INDUSTRY INTERNSHIP REPORT

On

“Python & Data Science”

By

C. YASWANTH (219X1A05B5)

Submitted in accordance with the requirement for the degree of B-Tech in

Department of Computer Science and Engineering

G. PULLA REDDY ENGINEERING COLLEGE (Autonomous): KURNOOL

(Affiliated to JNTUA, ANANTHAPURAMU)

KURNOOL - 518007

2024 – 2025

Department of Computer Science and Engineering

G. PULLA REDDY ENGINEERING COLLEGE (Autonomous): KURNOOL

(Affiliated to JNTUA, ANANTHAPURAMU)

CERTIFICATE

This is to certify that the internship titled "Python & Data Science" is a bonafide record of work carried out by C. Yaswanth (219X1A05B5), and the Internship report is submitted in partial fulfillment of requirements for the award of degree of Bachelor of Technology in Computer Science and Engineering.

………………………………..	………………………………..
INTERNSHIP GUIDE	**HEAD OF THE DEPARTMENT**
M. Shashidhar	**Dr. N. Kasiviswanath**
Assistant Professor	Professor & Head of the Department,
Department of C.S.E.,	Department of C.S.E.,
G. Pulla Reddy Engineering College,	G. Pulla Reddy Engineering College,
Kurnool.	Kurnool.

INTERNSHIP DETAILS

Name of the College: G. Pulla Reddy Engineering College.

Department: Computer Science and Engineering

Name of the Student: C. Yaswanth

Register Number: 219X1A05B5

Name of the Faculty Guide: M. Shashidhar

Duration of the Internship: From 06-01-2025 To 06-05-2025

Name and address of the Organization: Web Blinders, Kurnool.

Internship Domain: Data Science

Internship Title: Python and Data Science

Internship Stipend: N/A

Date of Internship Report Submission:

Internship Coordinator

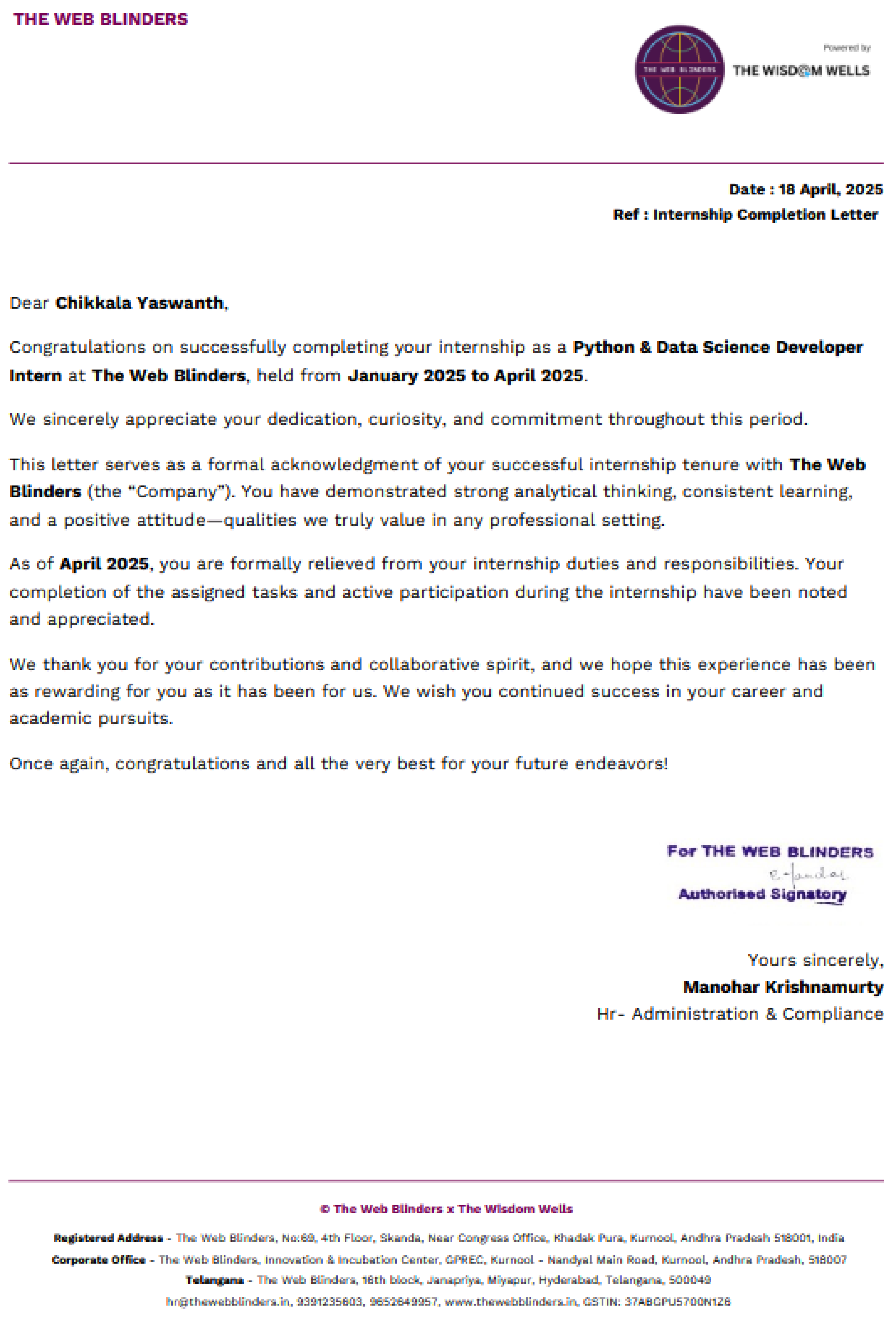
Dr. B. Geetha Vani
Professor
CSE Department
GPREC

DECLARATION

I, **C. Yaswanth**, a student bearing the Roll No. **219X1A05B5**, of the Department of Computer Science and Engineering, G. Pulla Reddy Engineering College, do hereby declare that the I have completed the mandatory internship titled "Python and Data Science Internship" from 06-01-2025 to 06-05-2025 in Web Blinders, Kurnool, under the Guide ship of Vineel Paridi**.**

(Student Signature and Date)

Internship completion certificate



THE WEB BLINDERS

Powered by
THE WISDOM WELLS

Date : 18 April, 2025
Ref : Internship Completion Letter

Dear **Chikkala Yaswanth**,

Congratulations on successfully completing your internship as a **Python & Data Science Developer Intern** at **The Web Blinders**, held from **January 2025 to April 2025**.

We sincerely appreciate your dedication, curiosity, and commitment throughout this period.

This letter serves as a formal acknowledgment of your successful internship tenure with **The Web Blinders** (the "Company"). You have demonstrated strong analytical thinking, consistent learning, and a positive attitude—qualities we truly value in any professional setting.

As of **April 2025**, you are formally relieved from your internship duties and responsibilities. Your completion of the assigned tasks and active participation during the internship have been noted and appreciated.

We thank you for your contributions and collaborative spirit, and we hope this experience has been as rewarding for you as it has been for us. We wish you continued success in your career and academic pursuits.

Once again, congratulations and all the very best for your future endeavors!

For THE WEB BLINDERS
[illegible]
Authorised Signatory

Yours sincerely,
Manohar Krishnamurty
Hr- Administration & Compliance

© The Web Blinders x The Wisdom Wells

Registered Address - The Web Blinders, No:69, 4th Floor, Skanda, Near Congress Office, Khadak Pura, Kurnool, Andhra Pradesh 518001, India
Corporate Office - The Web Blinders, Innovation & Incubation Center, GPREC, Kurnool - Nandyal Main Road, Kurnool, Andhra Pradesh, 518007
Telangana - The Web Blinders, 16th block, Janapriya, Miyapur, Hyderabad, Telangana, 500049
hr@thewebblinders.in, 9391235803, 9652849957, www.thewebblinders.in, GSTIN: 37ABCPU5700N126

ACKNOWLEDGEMENT

I wish to express my deep sense of gratitude to my internal supervisor **M.Shashidhar, Assistant professor** of Computer Science and Engineering Department, G. Pulla Reddy Engineering College.

I am thankful to our Internship Coordinator **Dr. B. GeethaVani** for her immaculate guidance, constant encouragement and cooperation which have made possible to bring out this internship work.

I am thankful to **Mr**. **Manohar Krishnamurty, HR - Administration & Compliance** of Web Blinders, for helping me and giving me the required information needed for my internship work.

I am thankful to our Head of the Department **Dr. N. Kasiviswanath**, for his whole hearted support and encouragement during the internship.

I am grateful to our Internship Dean **Dr. Y. V. Siva Reddy** and to our respected Principal **Dr. B. Sreenivasa Reddy** for providing requisite facilities and helping us in providing such a good environment.

I wish to convey my acknowledgements to all the staff members of the Computer Science and Engineering Department for giving the required information needed for my internship work.

Finally, I wish to thank all our friends and well-wishers who have helped me directly or indirectly during the course of this internship work.

Table of Contents

LIST OF FIGURES

1. INTRODUCTOIN

1. INTRODUCTION

1.1 ABOUT THE INTERNSHIP COMPANY/ORGANIZATION

A. Introduction of the Organization

Web Blinders, founded in 2018, is a rapidly growing software company and digital marketing firm in Kurnool, Andhra Pradesh. Focused primarily on offering innovative web solutions, Web Blinders has been at the forefront of empowering small and medium businesses (SMEs) to shift into the digital platform with strong website development, mobile applications, and online marketing strategy services.

The organization offers a wide variety of services, such as Website Design and Development, Android App Development, Search Engine Optimization (SEO), Social Media Marketing (SMM), Graphic Design, E-commerce Development, and Branding Solutions. Web Blinders adopts a customer-centric approach by providing bespoke solutions as per business requirements, with quality delivery and robust digital presence.

Web Blinders, with able leadership provided by experienced founders and a passionate team, has assisted many companies across various sectors like education, retail, healthcare, and real estate. Continuous learning, being agile, and delivering value via tech-based approaches are the thrusts of the team.

With a reputation for on-time delivery and innovation, Web Blinders is positioning itself as a go-to tech partner in the southern part of India by empowering companies with cutting-edge web technologies and scalable marketing solutions.

B. Vision, Mission and Values of the Organization

Vision:

To emerge as a top digital transformation service provider for upcoming businesses in India by leveraging the strength of technology, creativity, and strategy to develop digital success stories.

Mission:

To design scalable, responsive, and effective web and mobile solutions that enable businesses to excel in the digital age. Web Blinders aims to blend technical knowledge with creative thinking to provide result-driven services that drive clients' growth and visibility.

Core Values:

- **Innovation:** Continuously seeking new technologies and innovative approaches to provide distinctive solutions.
- **Integrity:** Creating long-term relationships through honesty and ethical conduct.
- **Customer Focus:** Providing customized solutions that address actual problems in the real world for customers.
- **Collaboration:** Encouraging teamwork and knowledge sharing to achieve success together.
- **Excellence:** Pursuing high quality in design, development, and delivery.

C. Policy of the Organization in Relation with the Intern Role

Web Blinders provides organized internship programs designed to provide students with practical experience in web technologies and online marketing methodologies. Interns are placed in existing client projects, which allow them to apply class concepts in live situations.

The internship program focuses on experiential learning, guidance by senior developers and marketers, and experience of end-to-end project life cycles. Interns are motivated to work across different areas such as front-end development, back-end scripting, UI/UX design, SEO, content creation, and client communication.

Web Blinders respects the growth of the intern, providing flexible learning spaces and feedback channels. Soft skills, ownership of tasks, and problem-solving are developed during the internship phase to equip students for the industry.

D. Organizational Structure

Web Blinders functions with a lean but effective team structure to facilitate quick decision-making and smooth project implementation:

- **Founders & Directors:** Vision-setting, strategic alliances, and finance.
- **Development Team:** Takes care of all technical functions, such as website and mobile app development.
- **Design Team:** Takes care of UI/UX, graphic design, and branding needs.

- **Marketing Team:** Implements SEO campaigns, handles social media platforms, and develops digital strategies.
- **Support & Admin:** Responsible for client coordination, HR functions, and overall administration.

This simplified organizational framework enables Web Blinders to provide customized services while ensuring agility and innovation in all projects.

E. Roles and Responsibilities of the Employees in Which the Intern is Placed

Interns at Web Blinders are assigned roles as per their academic background and professional interests. Their roles and responsibilities generally comprise:

1. **Web Development Assistance:** Helping to develop responsive websites with HTML, CSS, JavaScript, and front-end frameworks such as React or WordPress with the assistance of senior developers.
2. **Digital Marketing Support:** Providing assistance with SEO audits, keyword research, content creation, and social media campaign strategy and deployment.
3. **Design and UI/UX:** Working with the design team on layout compositions, visual elements, and interactive elements with the use of tools such as Figma or Adobe XD.
4. **Content Management:** Assisting in managing client sites, posting content, optimizing blog entries, and making mobile-friendly adjustments.
5. **Testing and QA:** Being involved in the testing process of websites and applications to detect bugs and offer improvement recommendations.

These activities ensure that interns receive valuable, practical experience, develop a solid technical base, and become more employable in the technology-based market.

1.2 VIRTUAL/OFFLINE INTERNSHIP DETAILS

Company Name: Web Blinders

Internship Role: Python & Data Science Intern

Team Mentors: Vineel Paridi

Projects Worked: Analysis and processing of data with statistical and graphical approach using python libraries.

Skills Acquired: Data Collection and Loading, Data Cleaning and Preprocessing, Exploratory Data Analysis (EDA), Data Visualization, Python Programming.

2. WORK DONE DURING INTERNSHIP

2. WORK DONE DURING INTERNSHIP

2.1 ABOUT COURSE/ DOMAIN

Introduction:

Python and Data Science Intern is a guided study and work program specifically designed for students and fresh graduates seeking exposure to real-world Python programming and the emerging arena of Data Science. The internship helps fill the gap between theoretical studies and industry requirements by providing experiential learning in data analysis, machine learning, data visualization, and actual project management.

Interns get hands-on experience in data collection, cleaning, analysis, and interpretation using Python libraries and tools and exposure to predictive modeling and business intelligence.

- **Python Programming:** Interns start with learning Python basics such as data types, control structures, functions, file handling, OOPs, exception handling, and modules, setting the stage for data science implementation.

- **Data Manipulation & Analysis:** Training involves utilizing libraries such as NumPy, Pandas, and OpenPyXL for tasks related to cleaning, transformation, and manipulation of data so that interns are able to gain insights from semi-structured and structured data.

- **Visualization:** Interns are trained to design static as well as interactive visualizations utilizing Matplotlib, Seaborn, and Plotly to inform storytelling and business reporting.

- **Statistics and Mathematics:** Fundamentals of probability, distributions, hypothesis testing, linear regression, and correlation are covered to enable scientifically interpreting data and aiding model development.

- **Machine Learning:** Interns are exposed to ML concepts through Scikit-learn with emphasis on supervised and unsupervised algorithms like linear regression, decision trees, clustering (K-Means), and model evaluation metrics.

- **Tools:** Familiarity with Jupyter Notebooks, Google Colab, Anaconda, and Git/GitHub is cultivated. Version control, virtual environments, and package management using pip/conda are stressed.

- **Processes:** Interns learn to adhere to CRISP-DM methodology (Cross-Industry Standard Process for Data Mining), which involves business understanding, data preparation, modeling, evaluation, and deployment.

- **People:** Interns work alongside data scientists, analysts, and project mentors, seeing how teams of people work together to answer real business questions with data.

- **Skills:** Focus is on analytical reasoning, coding effectiveness, statistical knowledge, and communication. Interns learn to communicate their findings and insights to non-technical audiences.

- **Organizations:** Industries such as e-commerce, healthcare, finance, education, and marketing accept Data Science interns who assist in trend analysis, customer segmentation, fraud detection, and others.

- **Methods:** Feature engineering, cross-validation, hyperparameter tuning, and pipeline development are the best practices adhered to for quality results.

- **Procedures:** Repetitive tasks involve data wrangling, exploratory data analysis (EDA), model training, result reporting, and documentation.

- **Rules:** Interns adhere to organization-defined data privacy, coding guidelines, and handling sensitive datasets.

- **Practices:** Experience with Agile Data Science, collaboration using Git, and mini-projects prepare interns for dynamic and high-intensity data-driven environments.

Stages of Learning & Contribution:

1. Fundamentals and Onboarding:

- Python Basics & Tools Setup: Variables, loops, functions, Python environments (Jupyter/Colab), and introduction to Git.

- Data Science Introduction: Overview of the discipline, case studies, and applications of real-world data.

2. Data Handling & Exploration:

- Data Wrangling: Pandas and NumPy for data cleaning, missing value handling, and EDA.
- Visualization Projects: Developing informative plots and dashboards for extracting insights.

3. Statistics and Modeling:

- Statistical Analysis: Interpretation of distributions, correlation, p-values, and application of statistical tests.
- ML Algorithms: Application of simple regression, classification models, and unsupervised learning using scikit-learn.

4. Real-World Projects & Collaboration:

- Domain-Specific Projects: Practical exercises such as sales forecasting, customer segmentation, or sentiment analysis.
- Team Work: Project briefings, model evaluations, and documentation of workflows.

5. Final Presentation & Review:

- Feedback & Certification: Mentor evaluations, feedback on performance, and issuance of internship certificate.

2.2 TECHNOLOGIES LEARNT DURING INTERNSHIP

1. Core Python:

- Essential programming concepts such as:
- Data types, conditionals, loops, functions, modules, and exception handling.
- OOP concepts (classes, objects, inheritance).
- File I/O and JSON/CSV/XML file processing.
- Application of built-in functions and libraries for utility development.
- Forms the basis for analytics, automation, and web scraping.

2. Data Science Libraries & Tools:

- **NumPy:** Numerical computing efficiently using arrays and vectorization.
- **Pandas:** Data manipulation and preprocessing using Series and DataFrames.

- **Matplotlib & Seaborn:** Visualizing trends, relationships, and distributions.
- **Scikit-learn:** Applying regression, classification, clustering, and model testing.
- **Jupyter Notebooks/Colab:** Interactive coding, notes, and visual reporting.
- **SQL/MySQL:** Running queries, joins, aggregations, and data retrieval using Python connectors.
- **Git & GitHub:** Version control, repository management, and collaboration.

2.3 TASKS AND ASSIGNED DETAILS

Project: Analysis and processing of data with statistical and graphical approach using python libraries.

Tasks Assigned:

- The task is to load and understand the e-commerce transaction dataset containing 68,565 records using pandas, and perform basic exploration using functions like head(), info(), and describe() to get an overview of columns such as transaction_id, user_id, product_id, quantity, price, timestamp, payment_method, and total.
- The task is to clean the data by checking for and handling any missing values, ensuring correct data types, and validating ranges for numerical features like quantity (e.g., 1 or more) and price (e.g., non-negative).
- The task is to remove duplicate entries to maintain data integrity, and ensure consistency across categorical variables like payment_method before proceeding with analysis.
- The task is to analyze transaction and product distribution by categories such as payment_method, timestamp (e.g., day of week, month), and product_id using bar plots, pie charts, and histograms via Matplotlib, Seaborn, and Plotly.
- The task is to explore and visualize price and quantity distribution patterns using histograms, KDE plots, and box plots to uncover variations across payment methods and product IDs.

- The task is to generate frequency-based insights, such as the most common payment methods and most frequently purchased products, and visualize them using bar charts and word clouds for better interpretation.

- The task is to summarize key e-commerce insights, such as most frequent payment methods, top-selling products, user purchase patterns by payment type, and trends in transaction timestamps, to support data-driven decisions and business strategies.

3. PROJECT COMPLETED

3. PROJECT COMPLETED

3.1 PROJECT

Project name: Analysis and processing of data with statistical and graphical approach using python libraries.

Technologies Used: Python, Pandas, NumPy, Matplotlib, Seaborn, WordCloud

Problem Statement

In today's digital commerce era, vast quantities of transaction data are being created on a daily basis from online stores, marketplaces, and payment gateways. This raw data containing user IDs, product details, quantities, prices, timestamps, and payment methods seeks to transform business strategies, customer understanding, and operational efficiency. But without effective cleaning, exploration, and visualization of the data, these opportunities lie hidden. The e-commerce dataset, with more than 68,000 records, offers a chance to reveal insightful patterns in user behavior, product popularity, and transaction trends.

Goal

The objective of this project is to examine and visualize the e-commerce transaction dataset with Python's data analysis and visualization libraries. The major emphasis is on determining user demographics (if available), top-selling products and payment methods, user purchase patterns by payment type, and temporal trends in online transactions. Through statistical summaries and visual storytelling, the project aims to identify actionable insights that can facilitate data-driven decision-making in online retail and business planning.

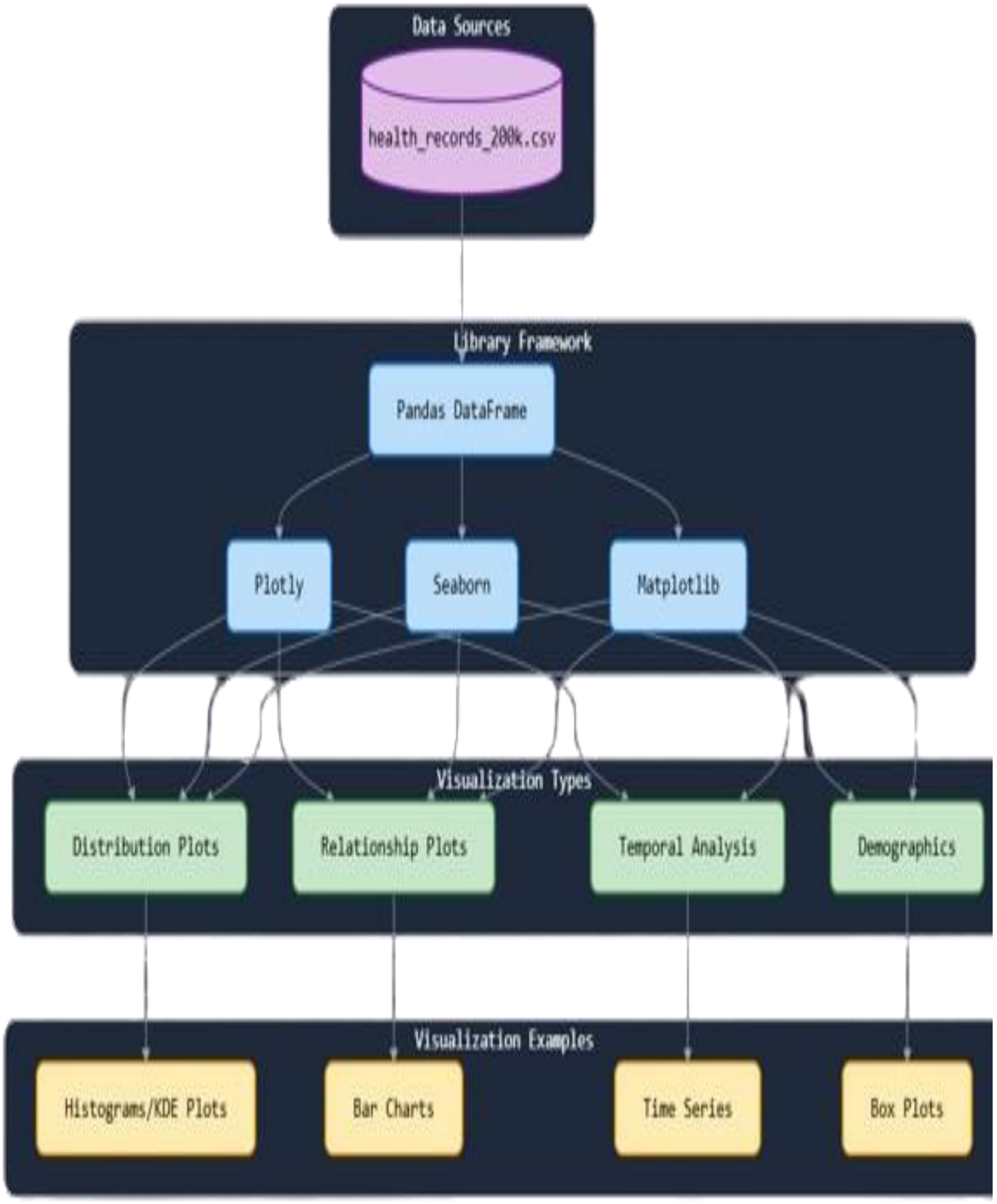


Fig 3.1.1 High Level Design

High-Level Design Overview

The architecture of the e-commerce analysis system consists of a four-layer pipeline which converts raw transaction data into insights. The system adheres to a structured process of data transformation—from ingestion, processing, and analysis to end-insight creation—wherein each layer plays a specific purpose in the journey of data.

System Architecture Components

1. Input Layer

The Input Layer is the first entry point for the system, which deals with the raw e-commerce dataset, including user IDs, product IDs, quantities, prices, timestamps, and payment methods. This layer ensures correct ingestion and formatting of the dataset, laying the groundwork for correct processing.

2. Processing Layer

The Processing Layer consists of three interconnected components:

- Data Loading Module
  - o Imports the e-commerce dataset using memory-efficient methods
  - o Validates data types and column integrity
  - o Ensures basic structure compliance for downstream processing
- Data Cleaning Module
  - o Handles missing values in user IDs, product IDs, quantities, prices, timestamps, and payment methods
  - o Standardizes entries for payment methods
  - o Ensures timestamp formatting for analysis
- Feature Engineering Module
  - o Extracts temporal features like day of week and month from timestamps
  - o Generates user-wise transaction counts and product purchase frequencies
  - o Calculates total purchase amount per transaction (already present as 'total')

3. Analysis Layer

The Analysis Layer performs deep data exploration through two specialized paths:

- Statistical Analysis Component
  - o Analyzes payment method distributions
  - o Evaluates product-wise purchase frequencies
  - o Tracks transaction trends over time
  - o Identifies top-selling products and common payment methods

- Data Visualization Component
    - o Generates bar charts for payment method distributions
    - o Creates pie charts for product category proportions (if product categories exist)
    - o Builds time-series plots for transaction volume trends
    - o Visualizes product-wise sales with bar graphs

4. Output Layer

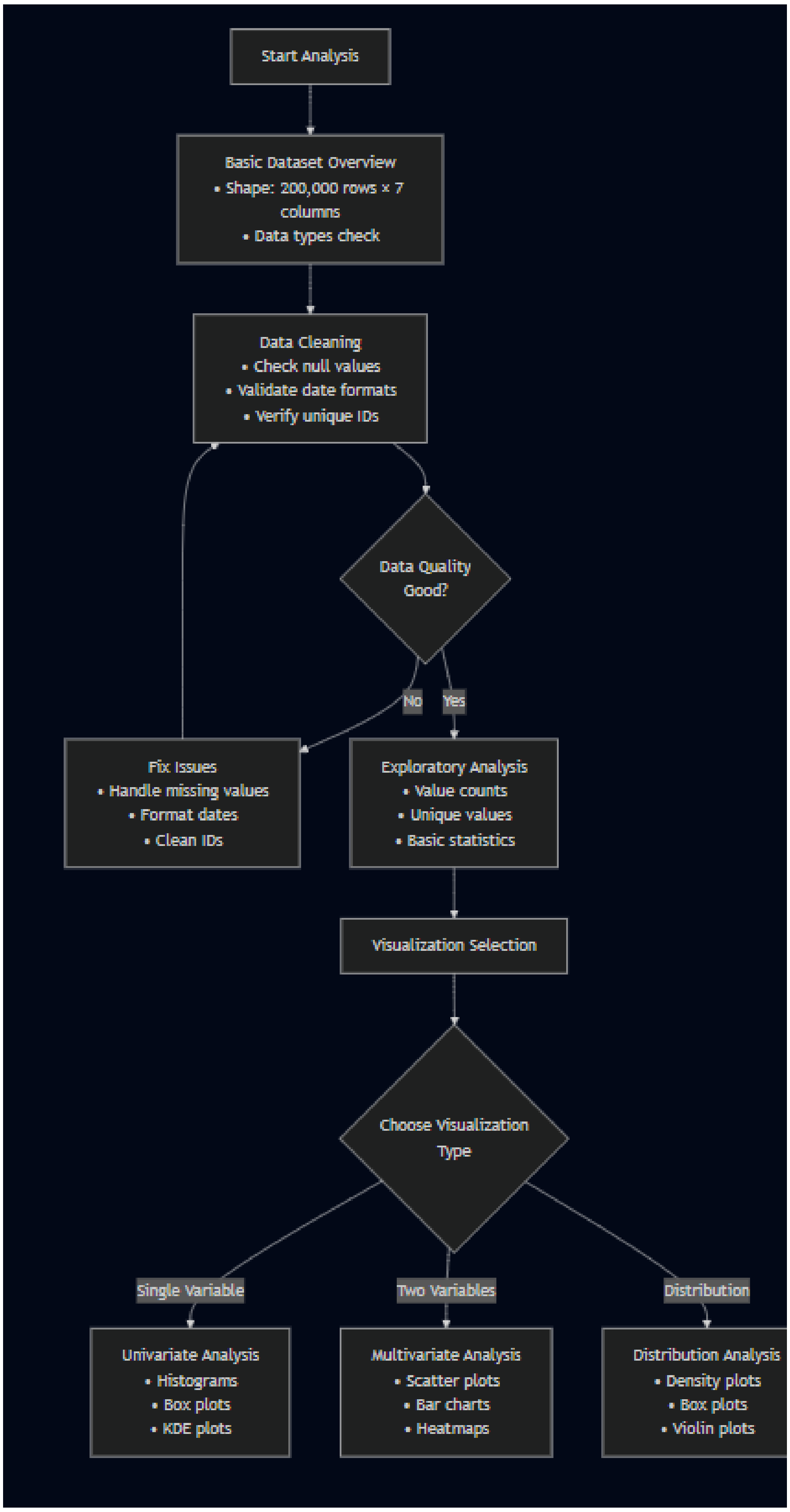
The Output Layer integrates findings from both analytical paths to deliver a complete picture of e-commerce trends, revealing:

- Payment method usage across transactions
- Product-wise popularity and sales volume
- Common purchase patterns and their frequency
- Transaction trends over time and potential seasonality

Data Flow and Integration

The end-to-end data flow within the ecommerce transactions analysis system is designed for seamless transition and insight generation:

1. Raw ecommerce records are imported through the Input Layer
2. Data undergoes cleaning and transformation in the Processing Layer
3. Refined data splits into Statistical and Visualization paths within the Analysis Layer
4. Insights are compiled in the Output Layer for reporting and interpretation.


Start Analysis
Basic Dataset Overview • Shape: 200,000 rows × 7 columns • Data types check
Data Cleaning • Check null values • Validate date formats • Verify unique IDs
Data Quality Good?
No
Yes
Fix Issues • Handle missing values • Format dates • Clean IDs
Exploratory Analysis • Value counts • Unique values • Basic statistics
Visualization Selection
Choose Visualization Type
Single Variable
Two Variables
Distribution
Univariate Analysis • Histograms • Box plots • KDE plots
Multivariate Analysis • Scatter plots • Bar charts • Heatmaps
Distribution Analysis • Density plots • Box plots • Violin plots

3.1.2 Project Flow

SOURCE CODE

1. Importing Required Libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plts

import seaborn as sns

import random

from wordcloud import WordCloud

The analysis begins by importing essential Python libraries such as pandas, numpy, matplotlib, seaborn, and WordCloud. These libraries are fundamental for handling, manipulating, and visualizing large-scale healthcare datasets effectively.

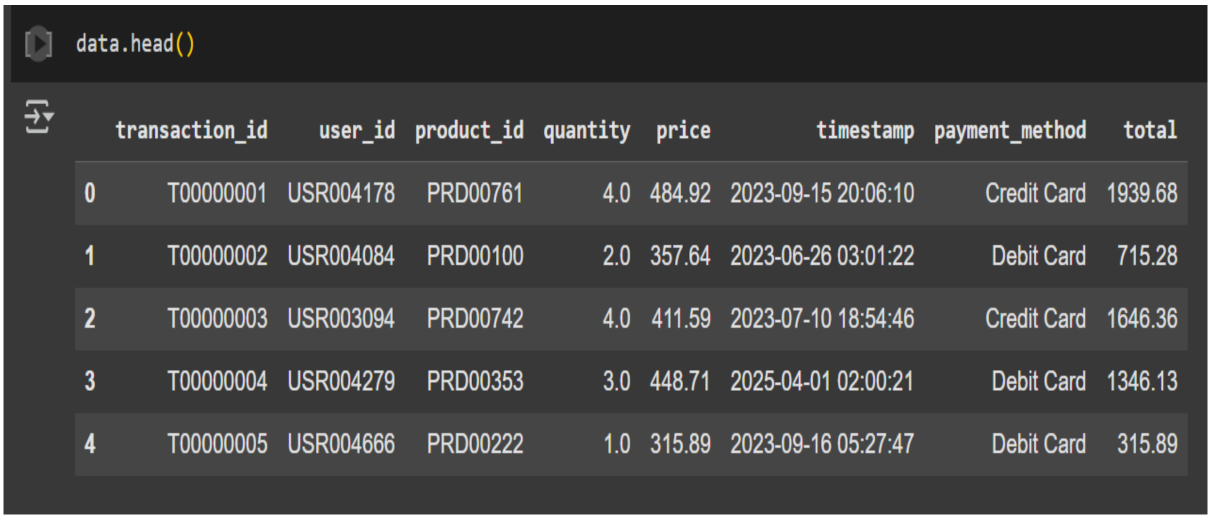
2. Loading and Displaying Dataset

data = pd.read_csv("ecommerce_transactions_200k.csv")

data.head()

data.tail()

The dataset containing 68,565 e-commerce transaction records is loaded using pd.read_csv(). The .head() and .tail() functions are used to examine the first and last few records of the dataset.

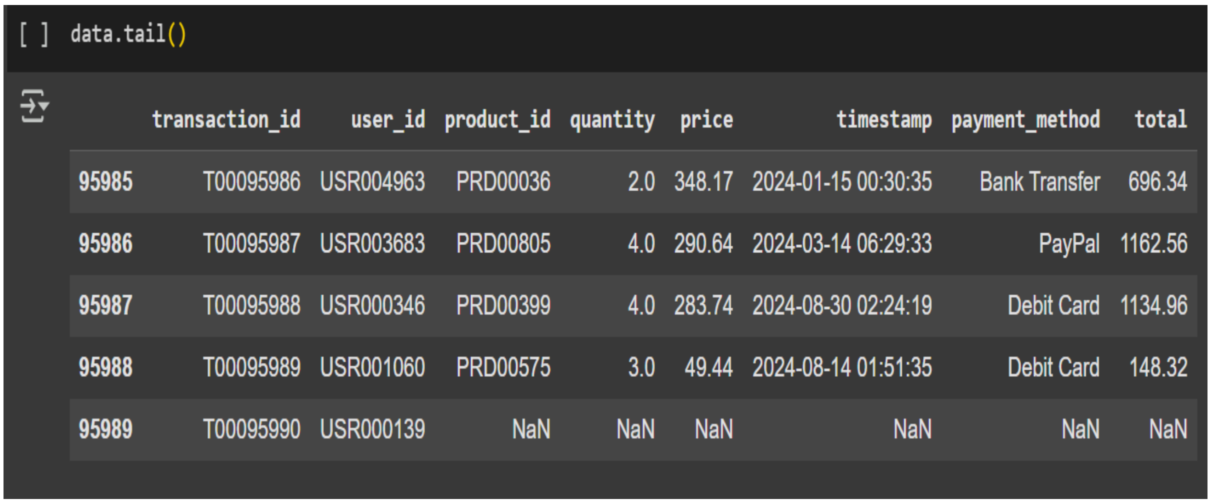


```
data.head()
```

```
  transaction_id    user_id product_id  quantity   price            timestamp payment_method    total
0      T00000001  USR004178   PRD00761       4.0  484.92  2023-09-15 20:06:10    Credit Card  1939.68
1      T00000002  USR004084   PRD00100       2.0  357.64  2023-06-26 03:01:22     Debit Card   715.28
2      T00000003  USR003094   PRD00742       4.0  411.59  2023-07-10 18:54:46    Credit Card  1646.36
3      T00000004  USR004279   PRD00353       3.0  448.71  2025-04-01 02:00:21     Debit Card  1346.13
4      T00000005  USR004666   PRD00222       1.0  315.89  2023-09-16 05:27:47     Debit Card   315.89
```

Fig 3.1.3 Displaying first 5 values

df.tail() will display the following values



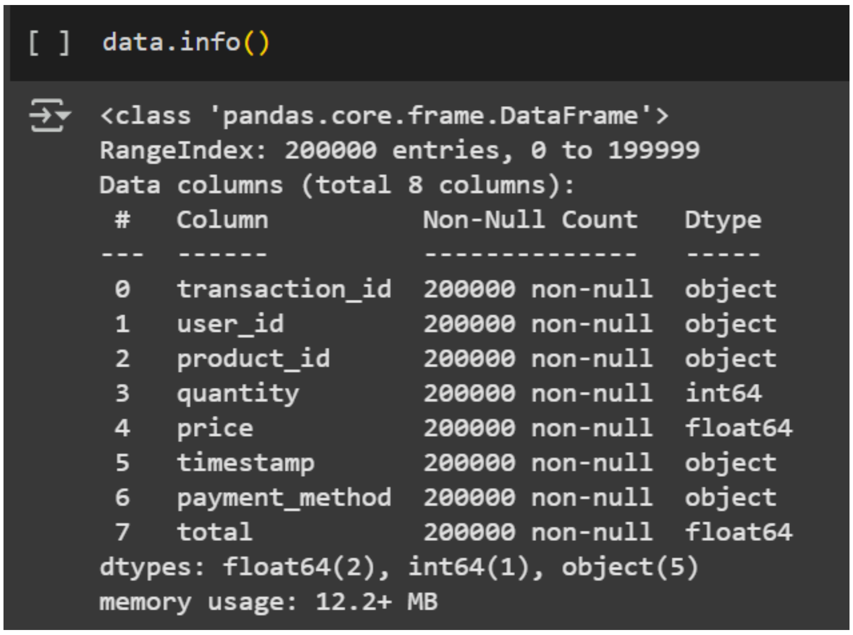
```
[ ] data.tail()

       transaction_id    user_id product_id  quantity   price            timestamp payment_method    total
95985       T00095986  USR004963   PRD00036       2.0  348.17  2024-01-15 00:30:35  Bank Transfer   696.34
95986       T00095987  USR003683   PRD00805       4.0  290.64  2024-03-14 06:29:33         PayPal  1162.56
95987       T00095988  USR000346   PRD00399       4.0  283.74  2024-08-30 02:24:19     Debit Card  1134.96
95988       T00095989  USR001060   PRD00575       3.0   49.44  2024-08-14 01:51:35     Debit Card   148.32
95989       T00095990  USR000139        NaN       NaN     NaN                  NaN            NaN      NaN
```

Fig 3.1.4 Displaying last 5 values

3. Initial Data Inspection

df.info()

It outputs the column names and dataset shape, helping to understand the structure and completeness of the dataset. This step is crucial for planning data cleaning.



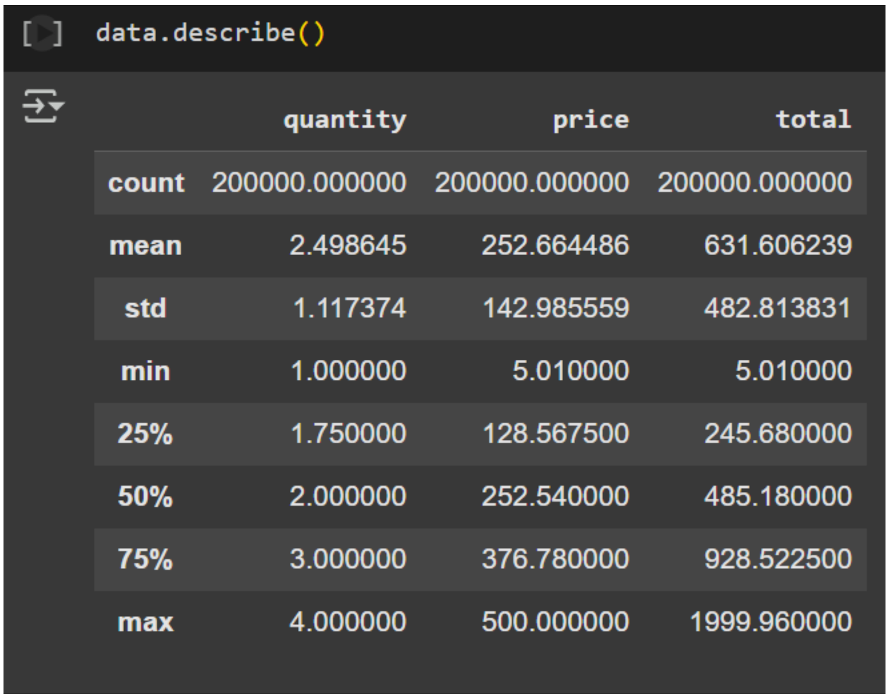
```
[ ] data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200000 entries, 0 to 199999
Data columns (total 8 columns):
 #   Column          Non-Null Count   Dtype  
---  ------          --------------   -----  
 0   transaction_id  200000 non-null  object 
 1   user_id         200000 non-null  object 
 2   product_id      200000 non-null  object 
 3   quantity        200000 non-null  int64  
 4   price           200000 non-null  float64
 5   timestamp       200000 non-null  object 
 6   payment_method  200000 non-null  object 
 7   total           200000 non-null  float64
dtypes: float64(2), int64(1), object(5)
memory usage: 12.2+ MB
```

Fig 3.1.5 Checking datatypes and non-null counts

4. Descriptive Statistics

df.describe()

Using df.describe(), a statistical summary of age column is generated, showing count, mean, standard deviation, and quartile values. This gives a sense of the distribution and range of age in the dataset.



```
data.describe()
```

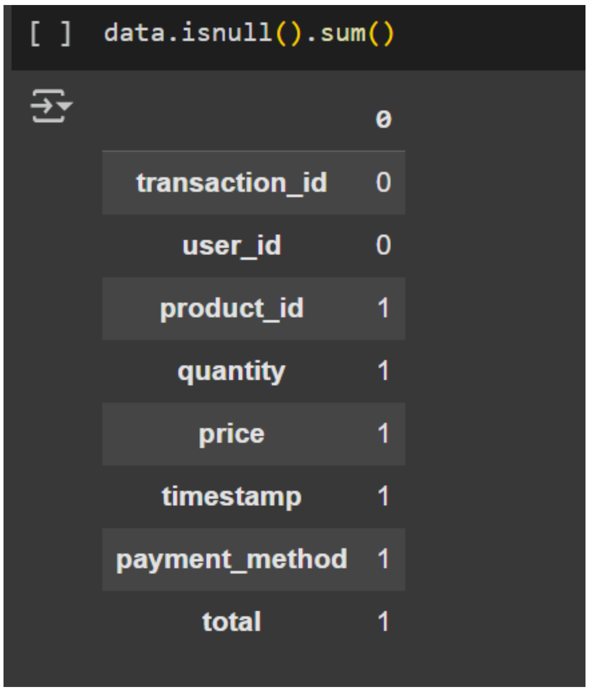
	quantity	price	total
count	200000.000000	200000.000000	200000.000000
mean	2.498645	252.664486	631.606239
std	1.117374	142.985559	482.813831
min	1.000000	5.010000	5.010000
25%	1.750000	128.567500	245.680000
50%	2.000000	252.540000	485.180000
75%	3.000000	376.780000	928.522500
max	4.000000	500.000000	1999.960000

Fig 3.1.6 price and quantity summary

5. **Checking Missing Values**

missing_values = df.isnull().sum()

The .isnull().sum() function identifies how many missing values exist in each column. This is important for determining where data cleaning is needed to handle incomplete entries before performing any analysis.



```
data.isnull().sum()
```

	0
transaction_id	0
user_id	0
product_id	1
quantity	1
price	1
timestamp	1
payment_method	1
total	1

Fig 3.1.7 Checking missing values

6. Analyzing Unique and Frequent Entries and Filtering Specific Medication Data

data['payment_method'].unique()

data['payment_method'].nunique()

data[data['payment_method'] == "Credit Card"]

The payment_method column contains 4 unique methods, including Credit Card, Debit Card, and PayPal. The third line filters all transaction records where Credit Card was used. This allows targeted analysis of payment preferences or related financial patterns.

7. **Visit Date Overview**

data['timestamp'].nunique()

data['timestamp'].unique()

The dataset contains 68,564 unique timestamps, spanning multiple years—indicating continuous and varied e-commerce activity.

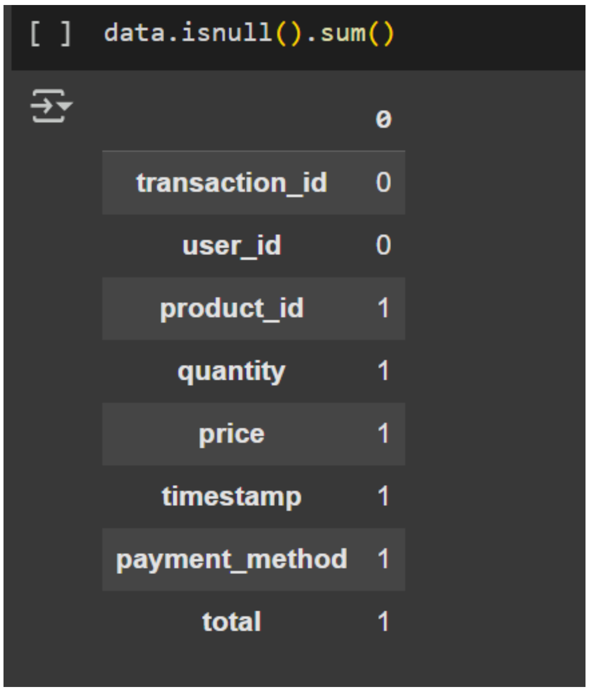
8. Final Check for Missing and Duplicates

print("Missing values after handling:\n", df.isnull().sum())

print("Duplicate rows:", df.duplicated().sum())

df = df.drop_duplicates()

After handling all missing values, the code verifies there are no remaining nulls and checks for duplicate rows. Any duplicate entries are removed to ensure the dataset is clean and reliable for analysis.



```
[ ] data.isnull().sum()
```

	0
transaction_id	0
user_id	0
product_id	1
quantity	1
price	1
timestamp	1
payment_method	1
total	1

Fig 3.1.8 Checking for missing values and duplicates

9. Distribution of payment methods

```
payment_counts = data['payment_method'].value_counts()
plt.figure(figsize=(8, 6))
payment_counts.plot(kind='bar', color=['#f9cdac', '#f2a49f', '#ec7c92', '#7db9e8', '#88d8b0'])
plt.title('Distribution of Payment Methods', fontsize=14)
plt.xlabel('Payment Method', fontsize=12)
plt.ylabel('Number of Transactions', fontsize=12)
plt.xticks(rotation=45)
plt.show()
```

The histogram reveals how price is distributed across the dataset, showing peaks and purchase value skews (e.g., more transactions with lower prices).

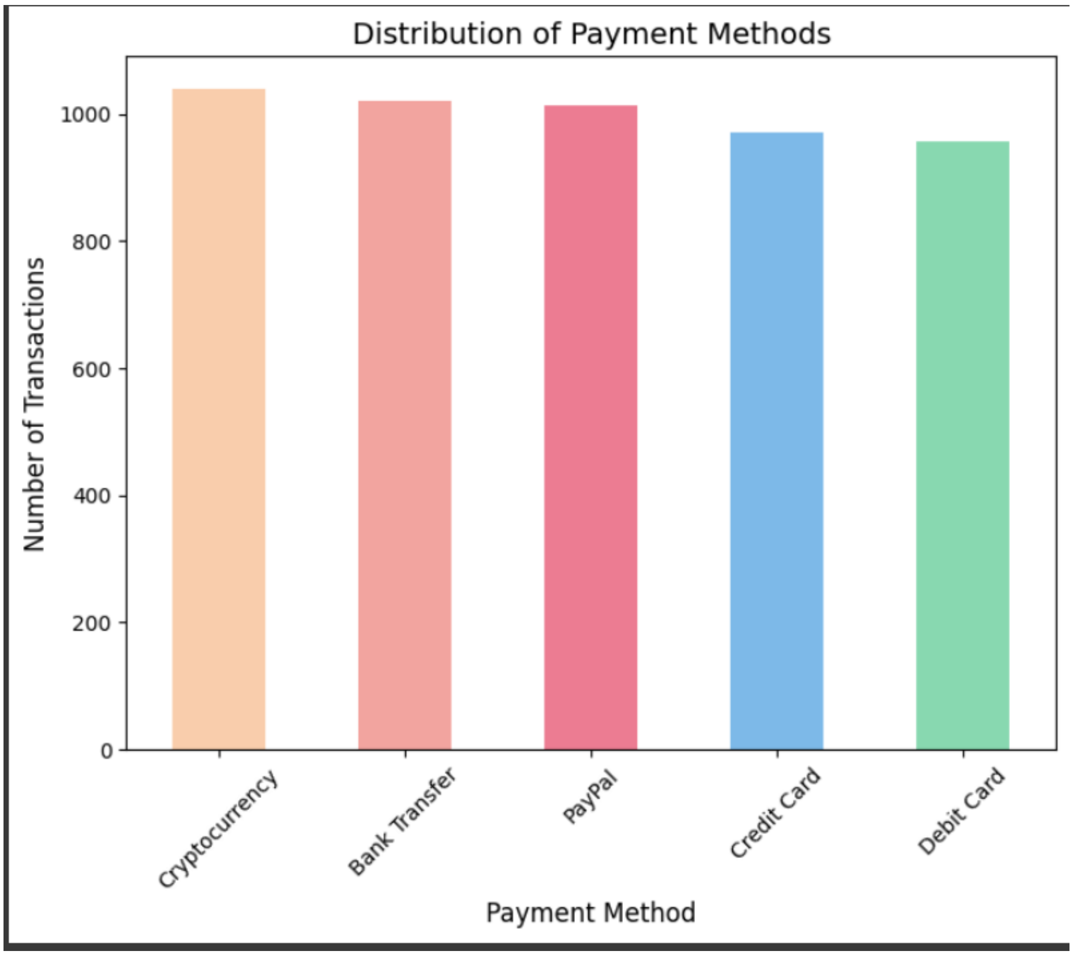


Fig 3.1.9 Bar graph for Payment method Distribution

10. Top 10 customers by Revenue

```
top_customers = data.groupby('user_id')['total'].sum().sort_values(ascending=False)[:10]

plt.figure(figsize=(8, 6))
top_customers.plot(kind='barh', color='#88d8b0')
```

```
plt.title('Top 10 Customers by Revenue', fontsize=14)
plt.xlabel('Total Revenue', fontsize=12)
plt.ylabel('User ID', fontsize=12)
plt.show()
```

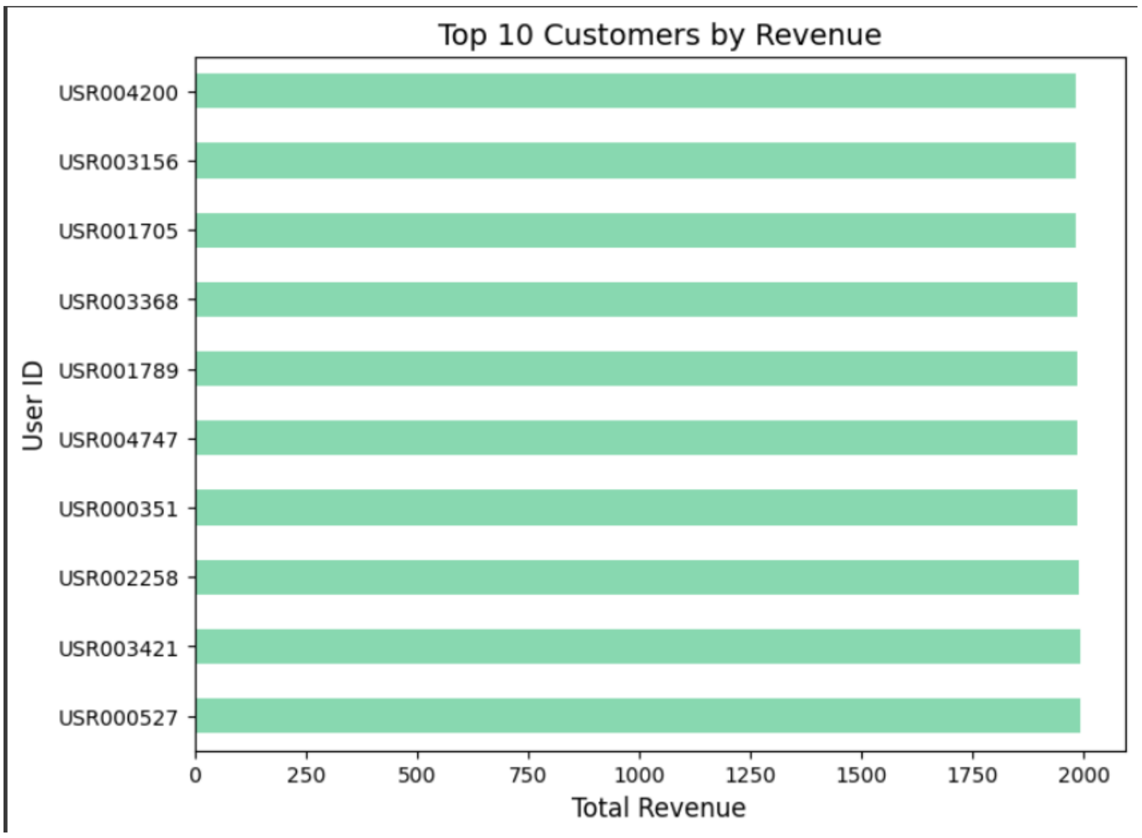


Fig 3.1.10 Bar chart for top 10 customers

11. Payment methods Distribution

```
payment_counts = data['payment_method'].value_counts()
plt.figure(figsize=(6,6))
plt.pie(payment_counts,
labels=payment_counts.index, autopct='%1.1f%%', colors=['#ff9999', '#66ffcc', '#66b3ff',
'#ffcc99', '#c2c2f0'])
plt.title("Distribution of Payment Methods")
plt.show()
```

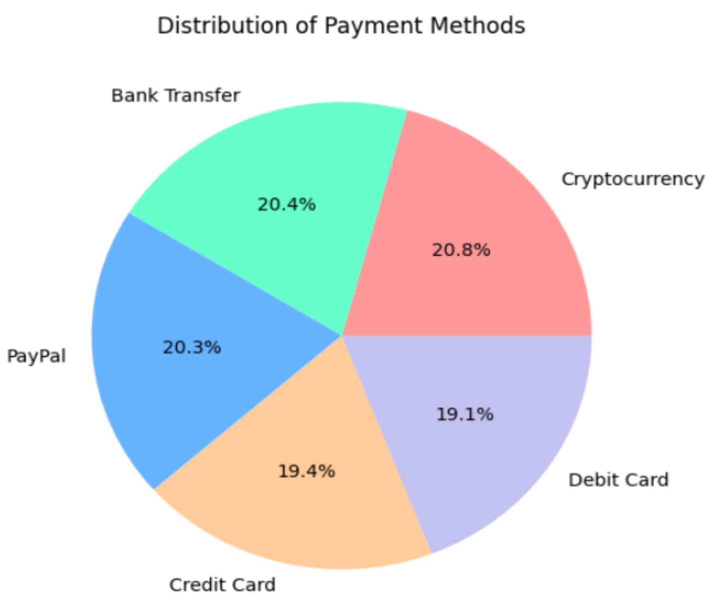


Fig 3.1.11 Pie chart for payment methods

12. Total Amount Denisty plot

```
if 'total' in data.columns:
    plt.figure(figsize=(8, 5))
    sns.kdeplot(data['total'], fill=True, color='skyblue')
    plt.title('Total Transaction Amount Density Plot')
    plt.xlabel('Total Transaction Amount')
    plt.ylabel('Density')
    plt.show()
```

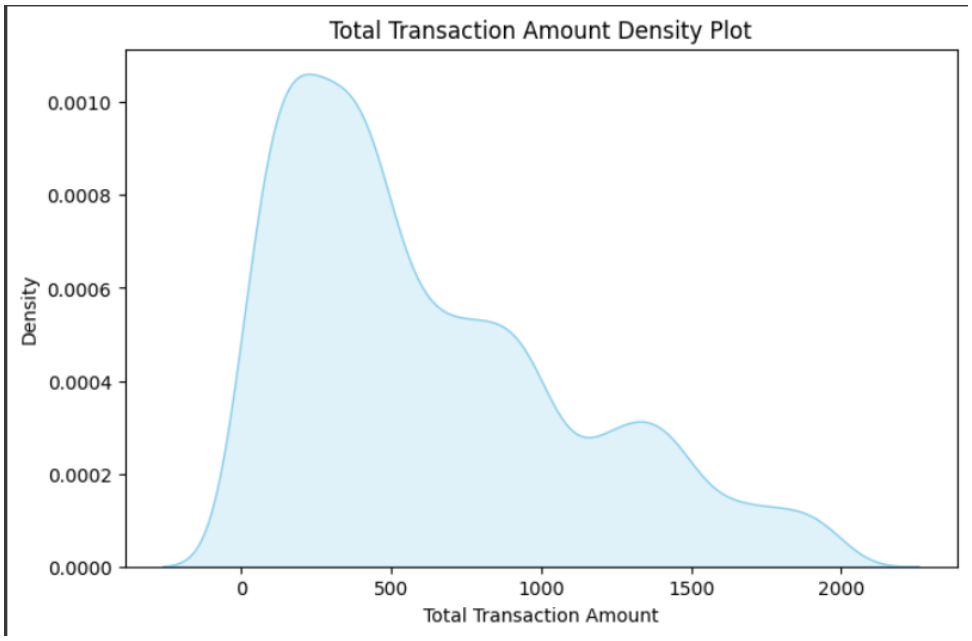


Fig 3.1.12 Total Amount Density Plot

3.2 LIST ASSIGNMENT

1. Remove empty strings from the list1 of strings

```
list1 = ["Mike","","Emma","Kelly","","Brad"]
list2 = []
for i in list1:
   if(len(i)!=0):
      list2.append(i)
list1 = list2
print(list1)
```

Output:

```
['Mike', 'Emma', 'Kelly', 'Brad']
```

This program iterates through each element in list1, checks if the element is not an empty string (""), and appends it to a new list list2. Finally, it replaces list1 with list2, which contains only non-empty strings.

2. **Add 7000 after 6000 in a nested list**

```
list1 = [10, 20, [300, 400, [5000, 6000], 500], 30, 40]
list1[2][2].append(7000)
print(list1)
```

Output:

```
[10, 20, [300, 400, [5000, 6000, 7000], 500], 30, 40]
```

This accesses the nested list [5000, 6000] using list1[2][2] and appends 7000 to it. This updates the inner list within the larger nested structure without affecting the rest of the list.

3. Replace first occurrence of 20 with 200

```
list1 = [5, 10, 15, 20, 25, 50, 20]
for i in range(len(list1)):
   if list1[i] == 20:
      list1[i] = 200
      break
print(list1)
```

Output:

```
[5, 10, 15, 200, 25, 50, 20]
```

This code loops through the list and checks for the first occurrence of the value 20.

Once found, it replaces it with 200 and exits the loop using break, so only the **first** occurrence is updated.

4. Remove all occurrences of a specific item (20)

```
list1 = [5, 20, 15, 20, 25, 50, 20]
list2 = []
for i in list1:
    if i != 20:
        list2.append(i)
list1 = list2
print(list1)
```

Output: [5, 15, 25, 50]

This program filters out all occurrences of the value 20 by checking each item and only adding the non-20 values to a new list list2. The original list is then updated with this filtered list.

5. **Concatenate two lists in a specific order**

```
list1 = ["Hello", "take"]
list2 = ["Dear", "Sir"]
result = []
for i in list1:
    for j in list2:
        result.append(i + " " + j)
print(result)
```

Output:

```
['Hello Dear', 'Hello Sir', 'take Dear', 'take Sir']
```

This uses nested loops to pair each string from list1 with each string from list2, forming a new combined phrase. Each combination is stored in a new list called result.

3.3 STRING ASSIGNMENT

```
string="OQYWFClFhFGAvIWYwGKpmZhnJiyzTslSIhSwvOsqJMEphzmifTkyqOMNpnOt
XZxmCfgDYqbaBHAUvIWhMnvwZnEMVDvmEfLrDoQnAZgQEgXQVnmSYkfedpAdhrt
pOgORpYLRZYGWdhWYuqQssCUXtTzKRDAhpjUheOzUroZNzWFtZOVwIapzUYtbSbj
YNErzQ"
```

1. Calculate the number of characters in the string

```
c = 0
for i in string:
    c += 1
print(c)
```

Output:

```
193
```

Explanation:

The loop iterates over every character in the string and increments the counter c. The final count is 193.

2. Count the number of occurrences of 'a' in the string

```
c = 0
for i in string:
    if i == 'a':
        c += 1
print(c)
```

Output:

```
2
```

Explanation:

This code checks each character in the string, and if it is 'a', it increases the count. There are exactly 2 'a' characters.

4. Return the substring starting at index 63 and ending at index 88 (inclusive)

```
print(string[63:89])
```

Output:

```
mCfgDYqbaBHAUvIWhMnvwZnEMV
```

Explanation:

Python string slicing is inclusive of the start index and exclusive of the end, so to include index 88, we slice till 89.

5. **Identify the character at index 45**

```
print(string[45])
```

Output:

```
z
```

Explanation:

Accessing string[45] directly gives the character at that position, which is 'z'.

3.4 STRING ASSIGNMENT - 2

1. Reverse a String

```
input = "hellohowareyoudoingandhowisyourhealth"
for i in range(len(input)-1, -1, -1):
    print(input[i], end="")
print()
```

Output:

```
htlaehruoysiwohdnagnioduoyerawoholleh
```

The loop starts from the last character (len(input)-1) and moves backward to the first character (-1 as the stop index, exclusive), printing each character in reverse. The result is the string printed backwards without using slicing.

2. Capitalize the First Letter of Each Word

```
input = "hello welcome to internship"
for i in input.split(" "):
    i = i[0].upper() + i[1:]
    print(i, end=" ")
print()
```

Ouptut:

```
Hello Welcome To Internship
```

This code splits the input string by spaces, capitalizes the first character of each word, and then prints them back in a sentence. It manually applies title-casing without using the .title() function.

3. Find the Longest Word in a Sentence

```
input = "he is very good at painting and paint good sketches"
max = ""
```

```
maxLen = 0
for i in input.split(" "):
    if maxLen < len(i):
        max = i
        maxLen = len(max)
print("The longest word in the sentence is : " + max)
print("with length : " + str(maxLen))
```

Output:

```
The longest word in the sentence is : painting with length : 8s
```

4. Length of the Last Word in a Sentence

```
input = "this is the most beautiful beach I ever visited"
s = ""
for i in input.split(" "):
    s = i
print("The length of the last word in the String is : " + str(len(s)))
```

Output:

```
The length of the last word in the String is : 7
```

The program iterates over each word, and the variable s gets updated with each new word. After the loop ends, s holds the last word. The length of this word is then printed.

5. Second Most Frequent Character in a String

```
input = "the hospital is very big and this hospital has so many doctors"
list = {}
for i in input:
    if list.get(i) == None:
        list[i] = 1
        continue
    else:
        list[i] += 1
max = 0
charMax = ''
secMax = 0
secCharMax = ''
```

```
for i in list.keys():
    if list.get(i) < max and list.get(i) > secMax:
        secMax = list.get(i)
        secCharMax = i
        continue
    elif list.get(i) >= max:
        secMax = max
        secCharMax = charMax
        max = list.get(i)
        charMax = i
print(secCharMax + " is the second most frequent Character in the string with frequency equal to " + str(secMax))
```

Output:

```
s is the second most frequent Character in the string with frequency equal to 7
```

Explanation:

1. The code counts the frequency of each character using a dictionary.
2. It then finds the character with the highest frequency (max) and the second-highest (secMax).
3. The second most frequent character and its count are printed.
4. Note: This also includes spaces in counting unless filtered.

3.5 STRING ASSIGNMENT – 3

1. Count the Number of Vowels in a String

```
input = "hasdfgeuionhjl"
s = "aeiou"
c = 0
for i in input:
    if i in s:
        c += 1
print("The no. of Vowels in the given string is : " + str(c))
```

Output:

```
The no. of Vowels in the given string is : 5
```

Explanation:

The code iterates through each character in the string and checks if it's a vowel by comparing it to a string containing all vowels (aeiou). A counter c is incremented for every vowel found.

2. **Count Occurrences of Each Word in a Sentence**

```
input = "hi hello hi and and hi more and more programming"
dic = {}
for i in input.split(" "):
    if dic.get(i) == None:
        dic[i] = 1
        continue
    dic[i] += 1
print(dic)
```

Output:

```
{'hi': 3, 'hello': 1, 'and': 3, 'more': 2, 'programming': 1}
```

Explanation:

The sentence is split into words using split(" "). Each word is added to a dictionary as a key, and its count is maintained as the value. This helps track the number of times each word appears.

3. String vs List in Python – 5 Differences

```
# 1) Type and Data Structure
s = "AbC"
l = ['A', 2, True]

print("1) "+s)            # String
print("1) "+str(l))       # List

# 2) Mutability
print("2) "+s.upper())    # Strings are immutable, returns a new string
print("2) "+s)            # Original string remains unchanged

l.append(5)               # Lists are mutable
print("2) "+str(l))

# 3) Indexing and Slicing
```

```
print("3) "+s[0:2])       # 'Ab'
print("3) "+str(l[0:2]))  # ['A', 2]

# 4) Concatenation
a = "ab"
b = "bc"
print("4) "+(a + b))      # String concatenation: 'abbc'

l = [1, 2, 3]
x = [1, 6, 7]
x = l + x
print("4) "+str(x))       # List concatenation: [1, 2, 3, 1, 6, 7]
# 5) Built-in Methods
# Strings: upper(), lower(), split(), replace()
# Lists: append(), pop(), extend(), sort()
```

Explanation:

Shows clear difference between strings and lists in terms of type, mutability, slicing, concatenation, and built-in methods.

4. Create Your Own String & Perform 5 Operations

```
s = "I am Thor god of Thunder and Loki is my brother from another mother."

# Operation 1: Uppercase
u_s = s.upper()
print(u_s)

# Operation 2: Length
length_s = len(s)
print("Length of the  string : ", length_s)

# Operation 3: Replace substring
new_string = s.replace("am", "a'm")
print(new_string)

# Operation 4: Split string
a = s.split()
print(a)
```

```
# Operation 5: Endswith check
b = s.endswith("mother")
print("Does the string end with mother? ", b)
```

OUTPUT

```
I AM THOR GOD OF THUNDER AND LOKI IS MY BROTHER FROM ANOTHER MOTHER.
Length of the  string :  68
I a'm Thor god of Thunder and Loki is my brother from another mother.
['I', 'am', 'Thor', 'god', 'of', 'Thunder', 'and', 'Loki', 'is', 'my', 'brother', 'from', 'another', 'mother.']
Does the string end with mother?  False
```

5. String Concatenation in Python

```
# Concatenation in strings means joining two or more strings together to form a single string.
f_n = "Arun"
l_n = "Sharon"
greeting = "Hello, "

full_greeting = greeting + f_n + " " + l_n + "!"
print(full_greeting)
```

Output:

```
Hello, Arun Sharon!
```

Explanation:

Using the + operator, three strings are concatenated intos one. Space and punctuation are added manually to make the final greeting readable.

3.6 PYTHON FUNCTIONS

1. Write a Python function to sum all the numbers in a list

```
# Function to find sum of all elements in a list
def findSum(x):
    ans = 0
    for i in x:
        ans += i
    return ans
num_tuple = (8, 2, 3, 0, 7)
print("The sum of all numbers in the given tuple is :", findSum(num_tuple))
```

Output:

```
The sum of all numbers in the given tuple is : 20
```

2. Write a Python function to multiply all the numbers in a list

```
# Function to multiply all elements in a list
def mul(x):
    ans = 1
    for i in x:
        ans *= i
    return ans
num_tuple1 = (8, 2, 3, -1, 7)
print("The answer after multiplying all the numbers in the given tuple is :", mul(num_tuple1))
```

Output:

```
The answer after multiplying all the numbers in the given tuple is : -336
```

3. Write a Python program to reverse a string

```
# Function to reverse a string
def reverse(s):
    ans = ""
    for i in range(len(s) - 1, -1, -1):
        ans += s[i]
    return ans
s = "1234abcd"
print("The reversed version of given string is :", reverse(s))
```

Output:

```
The reversed version of given string is : dcba4321
```

4. Write a Python function to calculate the factorial of a number

```
# Function to calculate factorial
def fact(x):
    ans = 1
    for i in range(1, x + 1):
        ans *= i
    return ans
```

```
x = int(input("Enter an input value: "))
print("The factorial of", x, "is :", fact(x))
```

Output (Example):

```
The factorial of 5 is : 120
```

5. Write a Python function to check whether a number falls within a given range

```
# Function to check if number falls in a given range
def check(x, i1, i2):
    if x >= i1 and x <= i2:
        return True
    else:
        return False
i1, i2 = [int(i) for i in input("Enter start and end of range (space-separated): ").split()]
x = int(input("Enter the number: "))

if check(x, i1, i2):
    print("The number", x, "falls in the range of", i1, "and", i2)
else:
    print("The number", x, "doesn't fall in the range of", i1, "and", i2)
```

Output (Example):

```
The number 20 doesn't fall in the range of 11 and 19
```

3. 7 PYTHON FUNCTIONS 2

1. Python function that accepts a string and counts the number of upper and lower case letters

```
# Function to count uppercase and lowercase letters in a string
def countLetters(s):
    ct = [0, 0]
    up = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    low = "abcdefghijklmnopqrstuvwxyz"
    for i in s:
        if i in up:
            ct[0] += 1
        elif i in low:
            ct[1] += 1
    return ct

s = "The quick Brow Fox"
ct = countLetters(s)
print("No.of Upper case characters :", ct[0])
print("No.of Lower case Characters :", ct[1])
```

Output:

```
No.of Upper case characters : 3
No.of Lower case Characters : 12
```

2. **Python function that takes a list and returns a new list with distinct elements from the first list**

```
# Function to return unique elements from a list
def getUnique(lst):
    s = []
    for i in lst:
        if i not in s:
            s.append(i)
    return s

num_list = [1, 2, 3, 3, 3, 3, 4, 5]
print("Unique List :", getUnique(num_list))
```

Output:

```
Unique List : [1, 2, 3, 4, 5]
```

3. Python function that checks whether a number is prime

```
# Function to check if a number is prime
def isPrime(x):
    if x < 2:
        return False
    for i in range(2, (x // 2) + 1):
        if x % i == 0:
            return False
    return True
```

```
x = int(input("Enter the input number: "))
if isPrime(x):
    print(x, "is a Prime Number")
else:
    print(x, "is not a prime number")
```

Output Example:

```
23 is a Prime Number
```

4. Python program to print the even numbers from a given list

```
# Function to return only even numbers from a list
def getEvenNumbers(lst):
    result = []
    for i in lst:
        if i % 2 == 0:
            result.append(i)
    return result

num_list11 = [1, 2, 3, 4, 5, 6, 7, 8, 9]
print(getEvenNumbers(num_list11))
```

Output:

```
[2, 4, 6, 8]
```

5. Python function to check whether a string is a pangram

```
# Function to check if a string is a pangram
def isPangram(s):
    arr = [False] * 26
    s = s.lower()
    for i in s:
        if i >= 'a' and i <= 'z':
            arr[ord(i) - 97] = True
    for i in arr:
        if i == False:
            return False
    return True

s = "The quick brown fox jumps over the lazy dog"
if isPangram(s):
    print("The given sentence is a pangram")
else:
    print("The given sentence is not a pangram")
```

Output:

```
The given sentence is a pangrams.
```

4. ACTIVITY LOG

4. ACTIVITY LOG FOR THE WEEK

weeks	Description of the weekly activity	Learning Outcome	Person In- Charge Signature
Week–1	Introduction to Python, variables, data types, loops, and conditionals. Practiced simple coding tasks.	Understood Python basics, syntax, and flow control using loops and conditional statements.	
Week-2	Explored Python functions, lists, tuples, dictionaries, and basic string manipulations.	Learned modular coding with functions and manipulation of data structures.	
Week-3	Practiced advanced string operations and list comprehensions.	Gained clarity on efficient string handling and concise list processing.	
Week-4	Learned file handling (read/write operations) and exception handling with try-except blocks.	Developed error-free programs involving file operations.	
Week-5	Solved small problems using functions: sum, product, reversing string, factorial, prime check, etc..	Strengthened problem-solving using logic, conditions, and function reuse.	
Week-6	Practiced range checking, character counting, substring extraction, and pangram checking using functions.	Built strong foundations in control structures and functional programming.	
Week-7	Focused on iteration-based problems, including even-number filtering and case-based letter counting.	Improved control flow, decision-making, and iteration logic.	

Week-8	Completed Python mini-programs combining all concepts. Prepared for data-focused libraries.	Gained confidence in core Python needed for real-world data applications.	
Week-9	Introduction to NumPy. Explored array creation, slicing, indexing, and operations.	Understood numerical data manipulation with high performance.	
Week-10	Introduction to Pandas. Worked with Series and DataFrames using head(), info(), and describe().	Learned to load, explore, and understand structured datasets.	
Week-11	Performed data cleaning using isnull(), fillna(), drop_duplicates(), and type conversions.	Gained essential skills in preprocessing real-world data.	
Week-12	Practiced filtering, grouping, and aggregation using Pandas. Applied custom operations..	Mastered transformation and aggregation of data using Pandas tools.	
Week -13	Visualized data using Matplotlib: line plots, bar charts, pie charts, and histograms.	Learned to represent data insights using clear visualizations.	
Week - 14	Created advanced visuals with Seaborn: heatmaps, countplots, boxplots, and distributions.	Enhanced ability to convey trends and patterns visually.	
Week – 15	Started Ecommerce Transactions dataset project: cleaned missing values, explored content types, ratings, and countries.	Implemented full cleaning and exploration workflow on real dataset.	
Week - 16	Finalized visual and statistical analysis of top payment methods, product popularity, and transaction trends over time.	Completed an end-to-end data analysis project using NumPy, Pandas, Matplotlib, and Seaborn.	

5.CONCLUSION

5. CONCLUSION

Looking ahead, there is immense potential to further develop and expand the facilities of data analysis and processing using the excellent collection of libraries offered by Python. Through this project, it has been shown how statistical and graphical techniques can be combined such that useful information is extracted, understood, and presented from actual datasets, such as the e-commerce transaction dataset. With the use of libraries like Pandas, NumPy, Matplotlib, and Seaborn, complex data is translated into powerful visual narratives that support knowledgeable decisions and trend identification.

The Python libraries' efficiency and flexibility save significant time and effort required for data visualization, exploration, and cleaning and make it ideal for students, researchers, and professionals. The project's findings—ranging from popular payment methods and product preferences to transaction trends over time—highlight the significance of statistical and graphical approaches in unveiling underlying patterns in raw data that would otherwise go unnoticed.

6.FUTURE ENHANCEMENTS

6. Future Enhancements

To extend further the reach and functionality of this project, using interactive visualization packages such as Plotly or Dash could allow the users a richer and more interactive experience. They would support live filtering and identification of trends, allowing the users to explore further the dataset based on their individual interests.

Moreover, incorporating machine learning models for content recommendation or binning similar countries and genres might add more analytical depth to the project. Predictive analytics can also be employed to forecast trends in content creation and user preference from historic data.

Finally, having this project initiated as an online-based dashboard would be simpler and more convenient to utilize by end users as they can review the findings without having to be programmers. Including user feedback features would also assist in guiding the implementation of new features as well as improve the overall user experience.

REFERENCES

1. Pandas Documentation – https://pandas.pydata.org/docs/
2. NumPy Documentation – https://numpy.org/doc/
3. Matplotlib Documentation – https://matplotlib.org/stable/contents.html
4. Seaborn Documentation – https://seaborn.pydata.org/
5. WordCloud Documentation – https://github.com/amueller/word_cloud
6. Python Official Documentation – https://docs.python.org/3/
7. Data Cleaning with Pandas – https://realpython.com/python-pandas-cleaning-data/
8. Data Visualization in Python – https://towardsdatascience.com/data-visualization-in-python-94a5b3cdd1c3
9. Exploratory Data Analysis (EDA) with Pandas and Python - https://www.datacamp.com/tutorial/exploratory-data-analysis-python
10. Kaggle: Python Data Science Handbook – https://www.kaggle.com/learn/pandas
11. Statistical Data Analysis in Python – https://realpython.com/python-statistics/
12. Visualizing Data with Seaborn – https://www.geeksforgeeks.org/python-data-visualization-with-seaborn/
13. Python Plotting With Matplotlib (Guide) – https://realpython.com/python-matplotlib-guide/
14. Kaggle Datasets – https://www.kaggle.com/datasets
15. Anaconda Distribution (NumPy, Pandas, Matplotlib bundled) – https://www.anaconda.com/